Mandare richieste AJAX con e senza JSON

```
let xhr = [];
    let numOfXHR = 2;
        for(let i = 0; i < numOfXHR; i++) {
            xhr[i] = new XMLHttpRequest();

            switch(i) {
                case 0:
                    xhr[i].open("POST", "S1", true);
                    xhr[i].setRequestHeader("Content-Type", "application/json");
                    break;
                case 1:
                    xhr[i].open("POST", "S2", true);
                    xhr[i].setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
                    break;
            }
        }

        for(let i = 0; i < numOfXHR; i++) {
            xhr[i].onreadystatechange = function() {
                if(xhr[i].readyState === 4 && xhr[i].status === 200) {
                    // Risposta ricevuta, esegui il codice qui
                    var response = xhr[i].responseText;

                    // S1 invia la risposta come stringa
                    // s2 invia la risposta come JSON

                    if(i == 0)
                        console.log("Risposta " + i + ": " + response);
                    if(i == 1) {
                        let jsObject = JSON.parse(response);
                        console.log("Risposta " + i + ": " + jsObject.id + " | " +
jsObject.value);
                    }
                }
            };

            // Invio ad S1 dati in formato JSON
            var dataForS1 = {
```

```
            "id": "This id is worth a lot",
            "value": "A lot"
        };

        console.log(dataForS1);
        let dataForS2 = "data1=" + text.value + "&data2=randomValue";

        if(i == 0)
            xhr[i].send(JSON.stringify(dataForS1));
        if(i == 1)
            xhr[i].send(dataForS2);
    }
```

Ricevere dati JSON (servlet)

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
    StringBuilder buffer = new StringBuilder();
    BufferedReader reader = request.getReader();
    String line;
    while ((line = reader.readLine()) != null) {
      buffer.append(line);
    }
    String data = buffer.toString();

    Gson gson = new Gson();
    MyObject parsedData = gson.fromJson(data, MyObject.class);

    System.out.println("Id: " + parsedData.getId());
    System.out.println("Value: " + parsedData.getValue());

    response.getWriter().print("I am S1");
}
```

Mandare dati JSON da servlet a client

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
        System.out.println("S2 | Received: " + request.getParameter("data1"));
        System.out.println("S2 | Received: " + request.getParameter("data2"));

        MyObject myObj = new MyObject("This is my id", "This is my value");
```

```
        Gson gson = new Gson();
        String json = gson.toJson(myObj);

        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        response.getWriter().write(json);
    }
```

Inserimento testo dopo tot. caratteri (no bottone invio)

```
// Inserimento testo e azione dopo tot caratteri (no bottone per l'invio)
let text = document.getElementById('txt');
text.addEventListener('keyup', (event) => {
    let numCaratteri = text.value.length;
    console.log("Caratteri inseriti: " + numCaratteri);
    if(numCaratteri == 100)
        alert("100 caratteri inseriti");
});
```

Controllo se il file è stato modificato in append (richiamare funzione checkFile(filePath)

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.*;

private static void checkFile(String path) {
        try {
            // Specifica il percorso del file da monitorare
            // TODO: cambiare percorso!
            Path fileToWatch = Paths.get(path);

            // Crea un oggetto WatchService
            WatchService watcher = FileSystems.getDefault().newWatchService();

            // Registra l'oggetto WatchService per ricevere notifiche di modifica
```

```
            fileToWatch.getParent().register(watcher, StandardWatchEventKinds.ENTRY_MODIFY);

            // Memorizza la dimensione iniziale del file
            lastSize = new File(fileToWatch.toString()).length();

            // Loop infinito per attendere eventi di modifica
            while (true) {
                WatchKey key = watcher.take();
                for (WatchEvent<?> event : key.pollEvents()) {
                    // Controlla se l'evento è di tipo ENTRY_MODIFY e se il nome del file
modificato corrisponde al file specificato
                    if (event.kind() == StandardWatchEventKinds.ENTRY_MODIFY &&
event.context().toString().equals(fileToWatch.getFileName().toString())) {
                        // Legge le nuove righe aggiunte al file
                        String newContent = readNewLines(fileToWatch.toString(), lastSize);
                        System.out.println("Modifiche al file " + fileToWatch.getFileName() + ":
");
                        System.out.println(newContent);
                        lastSize = new File(fileToWatch.toString()).length();
                    }
                }
                key.reset();
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
}

private static String readNewLines(String fileName, long lastSize) throws IOException {
        StringBuilder sb = new StringBuilder();
        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            // Salta le righe già lette
            br.skip(lastSize);
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line).append("\n");
            }
        }
        return sb.toString();
}
```

Scadenza sessione dopo tot minuti (5 min nell’esempio)

```
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class SessionExpirationServlet extends HttpServlet {
    private static final long SESSION_TIMEOUT = 5 * 60 * 1000; // 5 minuti in millisecondi

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        HttpSession session = request.getSession(true);

        // Imposta il timer di scadenza della sessione
        session.setMaxInactiveInterval((int) TimeUnit.MILLISECONDS.toSeconds(SESSION_TIMEOUT));

        // Controlla se la sessione è scaduta
        if (session.isNew() || session.getLastAccessedTime() < (System.currentTimeMillis() -
SESSION_TIMEOUT)) {
            // Notifica l'utente che la sessione è scaduta
            response.getWriter().println("La tua sessione è scaduta. Ti preghiamo di effettuare
nuovamente il login.");
            session.invalidate();
            return;
        }

        // Codice per la gestione della sessione normale
        // ...
    }
}
```

Servlet che stampa numero di sessione attive a console

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```java
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/SessionServlet")
public class SessionServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        HttpSession session = request.getSession();
        int activeSessions = session.getServletContext().getAttribute("activeSessions");
        System.out.println("Numero di sessioni attive: " + activeSessions);
    }
}
```

Prende in input una matrice e la divide in righe

```javascript
function splitNumbers() {
    var input = document.getElementById("numbers").value;
    var lines = input.split(",");
    var rows = [];
    for (var i = 0; i < lines.length; i++) {
        rows.push(lines[i].split(" "));
    }
    console.log(rows[0]);
}
```

Inserimento di tot caratteri (solo alfanumerici) e bottone disabilitato finchè non rispetta condizione

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Text Input App</title>
  </head>
  <body>
    <form>
      <label for="text-input">Insert text (100-1000 characters):</label>
      <input type="text" id="text-input" minlength="100" maxlength="1000">
```

```
      <button type="button" id="submit-button" disabled>Submit</button>
    </form>
    <script src="app.js"></script>
  </body>
</html>
```

```
const input = document.getElementById("text-input");
const submitButton = document.getElementById("submit-button");

input.addEventListener("input", () => {
    if (input.value.length >= 100 && /^[a-zA-Z0-9]+$/.test(input.value)) {
        submitButton.disabled = false;
    } else {
        submitButton.disabled = true;
    }
});

submitButton.addEventListener("click", () => {
    console.log(input.value);
});
```